

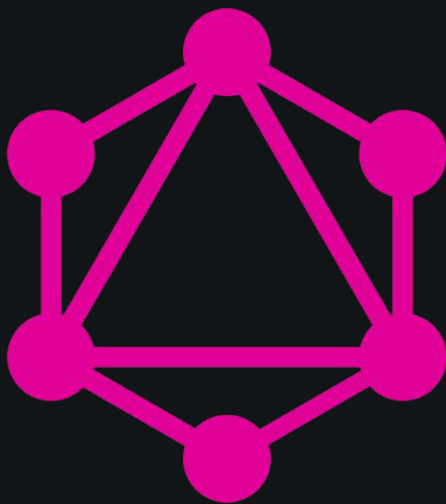
NODE.JS NIGHTS GRAPHQL

Josef Zavisek

STRV

AGENDA

- Introduction
- GraphQL language
- GraphQL server
- Demo



GraphQL

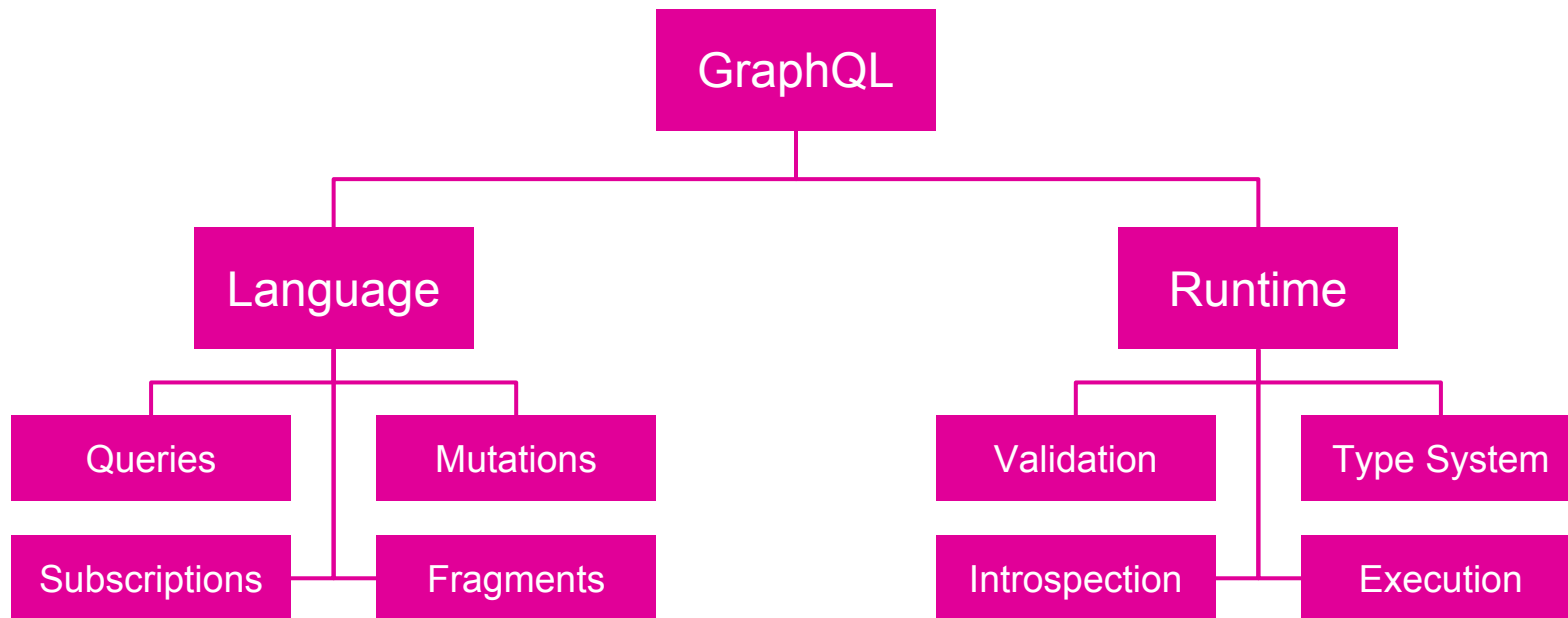
STRV

WHAT IS NOT GRAPHQL

- × Database
- × Query language for particular database technology
- × One specific implementation (apollo, express-graphql, graphql-ruby ...)

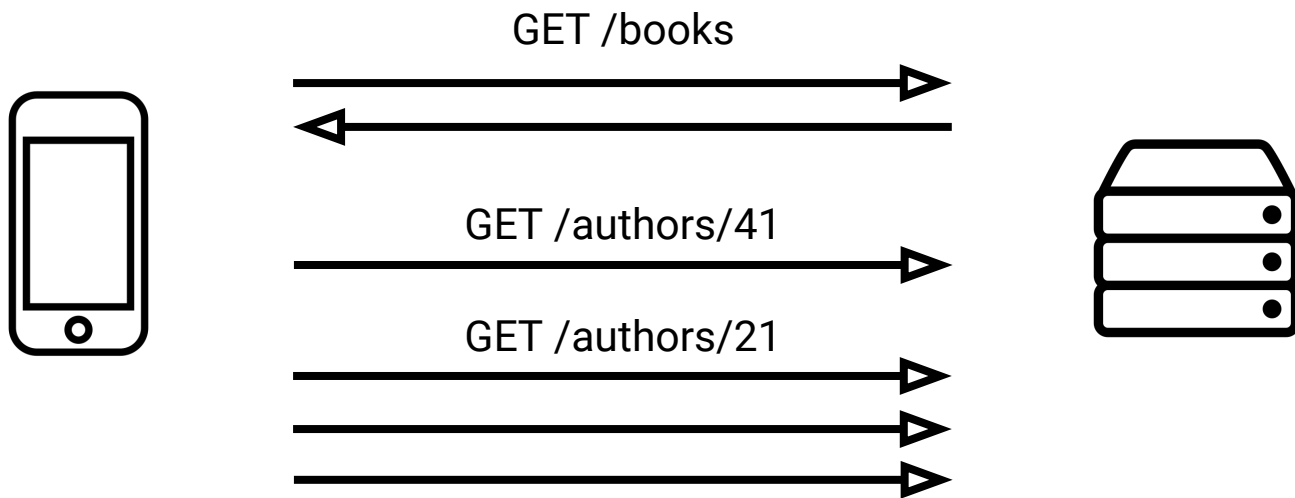
WHAT IS GRAPHQL

- Query language for APIs & server-side runtime for executing queries



WHAT'S WRONG WITH REST API?

- Too many round-trips to the server



- `GET /books?include=authors&authorFields=id,name&bookFields=name,gender`

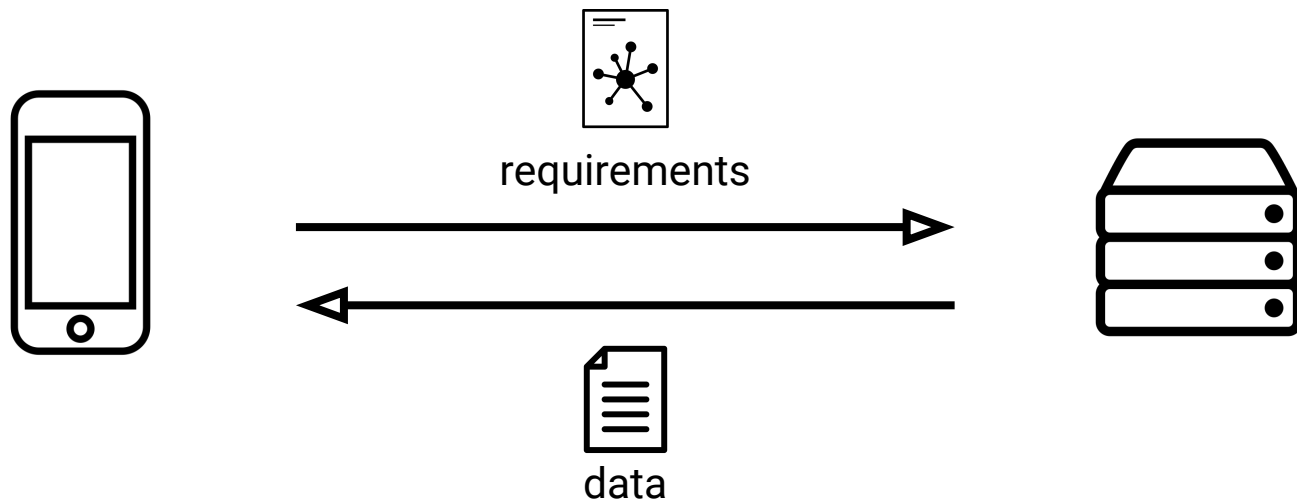
EXPENSIVE FIELDS

- Some fields might be hard to compute
- Overfetching
- Unpredictable shape of data

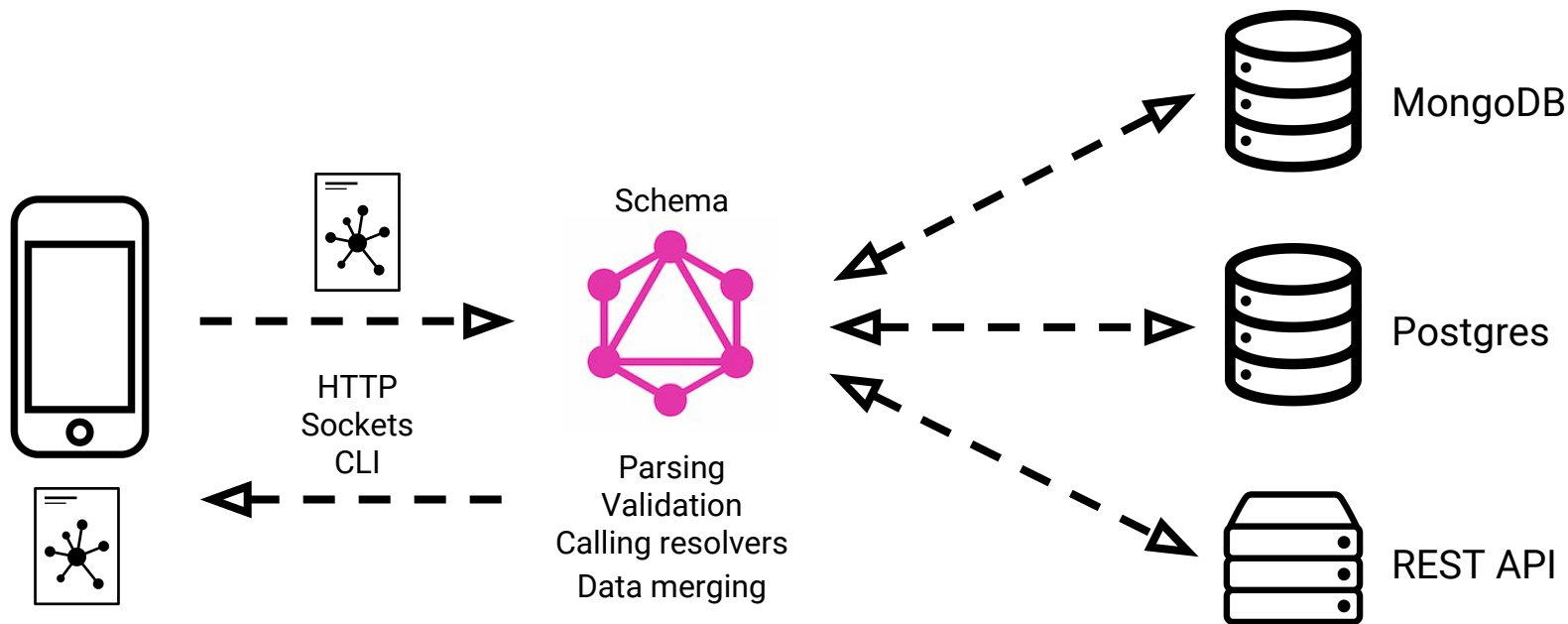
```
1  {
2    "books": [
3      {
4        "id": 1,
5        "name": "Yellow",
6        "genre": "Fiction",
7        "numberOfAwards": 8
8      }
9    ]
10 }
```

GRAPHQL SOLUTION

- Client specifies all the requirements at once



HOW DOES IT WORK?



GRAPHQL LANGUAGE

GRAPHQL LANGUAGE

```
1 query {  
2   dogs {  
3     id  
4     name  
5     breed  
6     birthYear  
7     photo  
8   }  
9 }  
  
1 {  
2   "data": {  
3     "dogs": [  
4       {  
5         "id": 1,  
6         "name": "Belle",  
7         "breed": "beagle",  
8         "birthYear": 2018,  
9         "photo": "https://images.dog.ceo/n02088364_10731.jpg"  
10      },  
11     {  
12       "id": 2,  
13       "name": "Dollie",  
14       "breed": "terrier",  
15       "birthYear": 2006,  
16       "photo": "https://images.dog.ceo/n02093754_3442.jpg"  
17     }  
18   ]  
19 }  
20 }
```

The diagram illustrates the mapping between a GraphQL query and its JSON response. The query on the left defines a field named 'dogs' which contains sub-fields: 'id', 'name', 'breed', 'birthYear', and 'photo'. The JSON response on the right shows a 'data' object with a 'dogs' array. Each element in the 'dogs' array is an object with the same fields as defined in the query. Arrows indicate the mapping: the 'dogs' field in the query maps to the 'dogs' array in the response; the 'id' field maps to the value '1'; the 'name' field maps to 'Belle'; the 'breed' field maps to 'beagle'; the 'birthYear' field maps to '2018'; and the 'photo' field maps to the URL 'https://images.dog.ceo/n02088364_10731.jpg'.

GRAPHQL LANGUAGE - QUERIES

- Name is optional
- Nested objects allowed

```
1  query getDogs {  
2    dogs {  
3      id  
4      name  
5      breed  
6      birthYear  
7      photo  
8      user {  
9        id  
10       name  
11     }  
12   }  
13 }
```

GRAPHQL LANGUAGE - QUERY VARIABLES

```
1  query getDogs($breed: String!) {  
2    dogs(filter: { breed: $breed }) {  
3      id  
4      name  
5      breed  
6      birthYear  
7      photo  
8      user {  
9        id  
10       name  
11     }  
12   }  
13 }
```

```
1  {  
2    "breed": "husky"  
3  }
```

GRAPHQL LANGUAGE - ALIASES

```
1 query getDogs {
2   belle: dog(id:1) {
3     id
4     name
5     breed
6     birthYear
7     photo
8   }
9   dollie: dog(id:2) {
10    id
11    name
12    breed
13    birthYear
14    photo
15  }
16 }
```

```
1 {
2   "data": {
3     "belle": {
4       "id": 1,
5       "name": "Belle",
6       "breed": "beagle",
7       "birthYear": 2018,
8       "photo": "https://images.dog.ceo/n02088364\_10731.jpg"
9     },
10    "dollie": {
11      "id": 2,
12      "name": "Dollie",
13      "breed": "terrier",
14      "birthYear": 2006,
15      "photo": "https://images.dog.ceo/n02093754\_3442.jpg"
16    }
17  }
18 }
```

GRAPHQL LANGUAGE - FRAGMENTS

```
1 query getDogs {  
2   belle: dog(id:1) {  
3     ... DogFields  
4   }  
5   dollie: dog(id:2) {  
6     ... DogFields  
7   }  
8 }  
9  
10 fragment DogFields on Dog {  
11   id  
12   name  
13   breed  
14   birthYear  
15   photo  
16 }
```

```
1 {  
2   "data": {  
3     "belle": {  
4       "id": 1,  
5       "name": "Belle",  
6       "breed": "beagle",  
7       "birthYear": 2018,  
8       "photo": "https://images.dog.ceo/n02088364\_10731.jpg"  
9     },  
10    "dollie": {  
11      "id": 2,  
12      "name": "Dollie",  
13      "breed": "terrier",  
14      "birthYear": 2006,  
15      "photo": "https://images.dog.ceo/n02093754\_3442.jpg"  
16    }  
17  }  
18 }
```

GRAPHQL LANGUAGE - DIRECTIVES

```
1  query getDogs($withUser: Boolean!) {  
2    dogs {  
3      id  
4      name  
5      breed  
6      birthYear  
7      photo  
8      user @include (if: $withUser) {  
9        id  
10       name  
11     }  
12  }  
13 }
```

```
1  {  
2    "withUser": true  
3  }
```


GRAPHQL LANGUAGE - MUTATIONS

```
1  mutation createDog($input: CreateDogInput!) {  
2    createDog(input: $input) {  
3      id  
4      name  
5      breed  
6      birthYear  
7      photo  
8    }  
9  }
```

```
1  {  
2    "input": {  
3      "name": "Dollie",  
4      "breed": "terrier",  
5      "birthYear": 2018  
6    }  
7  }
```

GRAPHQL SERVER

SERVER

- We will use “apollo-server-koa” package

```
• • // Create Apollo server
• • const server = new ApolloServer({
• •   typeDefs,
• •   resolvers,
• •   debug: true,
• •   introspection: true,
• •   context: makeContext,
• •   playground: playgroundConfig,
• •   engine: engineConfig,
• •   formatError,
• • })
```

```
• • // Apply Apollo middleware
• • server.applyMiddleware({
• •   app,
• •   path: '/graphql',
• • })
```

SCHEMA / TYPEDEFS

- Usually in .gql or .graphql files
- We can use “merge-graphql-schemas” package to load it

```
1  type User {  
2    id: Int!  
3    name: String!  
4  }  
5  
6  type Dog {  
7    id: Int!  
8    name: String!  
9    breed: String!  
10   birthYear: Int!  
11   photo: String  
12   age: Int!  
13   user: User  
14 }  
15  
16 type Query {  
17   dog(id: Int!): Dog  
18   dogs: [Dog]  
19 }
```

RESOLVERS

- Their structure has to match the schema

```
1  'use strict'
2
3  const operations = require(' ../ ../operations/dogs')
4
5  module.exports = {
6    Query: {
7      dog: (root, args) => operations.getById({ id: args.id }),
8      dogs: () => operations.getAll(),
9    },
10   Dog: {
11     // Computed field
12     age: dog => dog.age || new Date().getFullYear() - dog.birthYear,
13   },
14 }
```

LET'S CODE

QUESTIONS

STRV

THAT'S IT

Josef Zavisek / @jzavisek, @strvcom

STRV

STRV